# Table of Contents

## PART A



## PART B

# Abstract

Brain tumour is caused by an abnormal growth of cells around the brain that can affect the quality of life in a patient regardless of whether the tumour is cancerous or not. Hence, early detection and treatment is vital because the tumour may put pressure on other parts of the brain that control the vital functions of the body as the tumour grows, which can lead to other issues such as loss of vision, paralysis and vision or speech problems. Due to the complexities in identifying the types of brain tumours from MRI screenings, an automated system which can detect the presence of brain tumour followed by classifying them can be beneficial in the field of oncology. In this assignment, a deep learning model is developed to classify only three types of brain tumours, namely glioma, meningioma, and pituitary tumour provided if there is a tumour detected. A convolutional neural network based on the pre-trained model, EfficientNet-B0, is used to classify different classes of brain tumour. The tuned EfficientNet-B0 model obtained 99.43% testing accuracy, 99% precision, 99% recall, and a 99% F1-score. The performance of this model is on par with other models mentioned in the literature review such as the Wavelet-based CNN (WCNN) model and the "DeepTumourNet" model.

## 1.0 Introduction

Brain tumour is caused by a rapid growth of cells in the brain in an uncontrollable manner. According to Bhatt *et al.* (2022), brain cancer accounts for approximately 1.3% of all the cancer diagnoses in the US every year. All brain cancers are the result of brain tumour, but, not all brain tumours are cancerous. The non-cancerous tumours are being labelled as benign while the cancerous tumours are called malignant. The symptoms of having a brain tumour may vary depending on the location of the tumour growing inside the brain. Some of the common symptoms include seizures, behavioural or personality changes, progressive paralysis on one side of the body, vision or speech problems (National Health Service, 2021). Although the causes of brain tumours remains unknown, there are a few factors that may increase the risk of developing a brain tumour. Some of these factors are exposure to excessive radiation, aging and genetic conditions (Park et al., 2017).

Some of the common treatments of brain tumour includes surgery, radiotherapy, chemotherapy or a combination of any of these two depending on the situation. However, these treatments do not guarantee the complete removal of the brain tumour. According to Aldape et al. (2019), conventional brain tumour treatment like surgery are not effective against brain tumours that are located beyond the reach of neurosurgeon. As for tumours that that are located near the sensitive neural tissues, exposure to the radiation of chemotherapy is not feasible. Despite all this, early detection of the tumours followed by immediate treatment can significantly improve the chances of recovery of a patient.

Fortunately, with the advancement in technology such as machine learning/artificial intelligence, the task of detection and classification of brain tumours can be made more effective and efficient. Presented in this assignment is a deep learning based model based on Convolutional Neural Networks (CNN) paired with transfer learning approach to help detect the existence of brain tumour and further classify them into three types, namely glioma, meningioma and pituitary tumour.

## 1.1 Problem Statement

Currently, a Magnetic Resonance Imaging (MRI) screening is the best way to detect brain tumours. However, due to the to the level of complexities involved in brain tumours and their properties, the identification of the type of brain tumours are complicated to begin with. Hence,

a thorough analysis by professionals on the MRI images is required to determine whether the tumours are either malignant or benign. Occasionally, these manual examinations could be prone to error and bias as different professionals have different point of view based on their own experiences. Also, the lack of medical professionals in developing countries makes it difficult and time-consuming to determine the properties of the brain tumour based on the MRI screening. Hence, an automated brain tumour detection and classification system can be developed using deep learning algorithms such as Convolution Neural Network (CNN) to assist medical practitioners in their brain tumour diagnosis and treatment plans.

## 1.2 Aim

The aim of this research is to develop a deep learning-based detection and classification model to detect and classify the different types of brain tumors.

## 1.3 Objectives

As stated below are the objectives of this project:

1. To explore and understand the existing deep learning architectures in the application of identifying and classifying tumours
2. To develop deep learning-based models for the detection and classification of brain tumours
3. To enhance the performance of the model by performing fine tuning

## 1.4 Background

### 1.4.1 Deep Learning Application In Medical Image Analysis

The application of deep learning in medical image analysis have been widely adopted due to its accurate and efficient object detection, segmentation, classification of pathophysiological anatomical structures (Tsuneki, 2022). Besides supporting medical professionals in disease diagnosis of their patients, deep learning also allows them to make better decisions in terms of providing treatment to the patient.

However, the robustness of deep learning in medical image analysis is limited to the size of the provided training set, where sufficient medical images are not always available. According to Agrawal and Juneja (2019), the authors have highlighted that data scarcity is hindering the development of deep learning in medical imaging because experts have to annotate the data to prevent any human error, which is a time-consuming process. Also, class imbalance is also an

issue as there is a challenge in collecting samples of rare diseases. Hence, more research and efforts are required to develop robust deep learning-based computer-aided diagnosis applications for better clinical workflow in the medical field.

### 1.4.2 Different Types of Brain Tumours

Brain tumours can be classified into different classes based on the tumour location in the brain. Among all the other types of brain tumours, meningioma is the most common one, accounting for more than 30% of all brain tumours (American Association of Neurological Surgeons, 2022). Fortunately, these tumours are mostly benign, slow-growing tumours, and there is a very small chance of being cancerous tumours. Meningioma tumours grow from the meninges, which is the tissues that surround and protect the brain just right under the skull. As shown below are the samples of MRI images with meningioma.

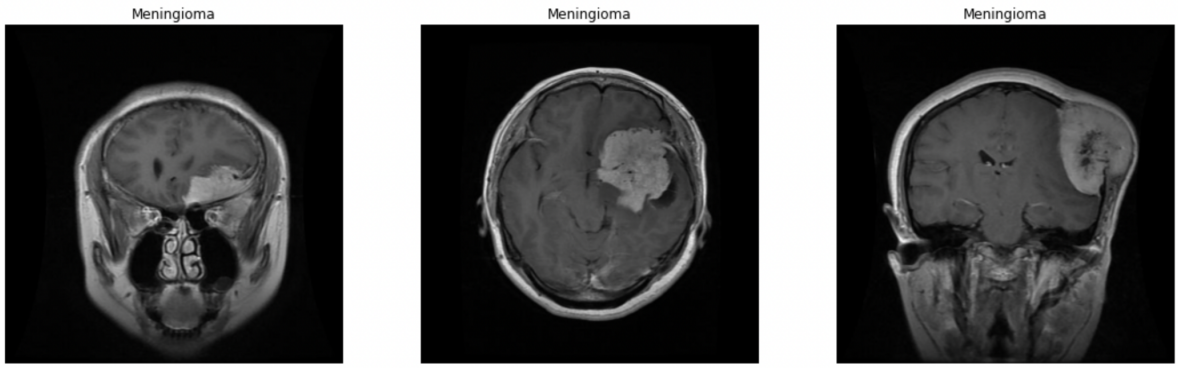



*Figure 1: Brain MRI Images with meningioma detected*

Glioma is another type of primary brain tumours. It is a tumour that starts out in the glial cells, which are supporting cells of the brain and spinal cord. When these glial cells grow uncontrollably, tumours will form typically near the spinal cord. Although gliomas are considered as cancerous, the tumours can grow at a very slow pace in some cases. Gliomas can be further classified into other types 3 types, namely astrocytomas, ependymomas, and oligodendrogliomas. As shown below are some of the samples of MRI images with glioma tumours detected.

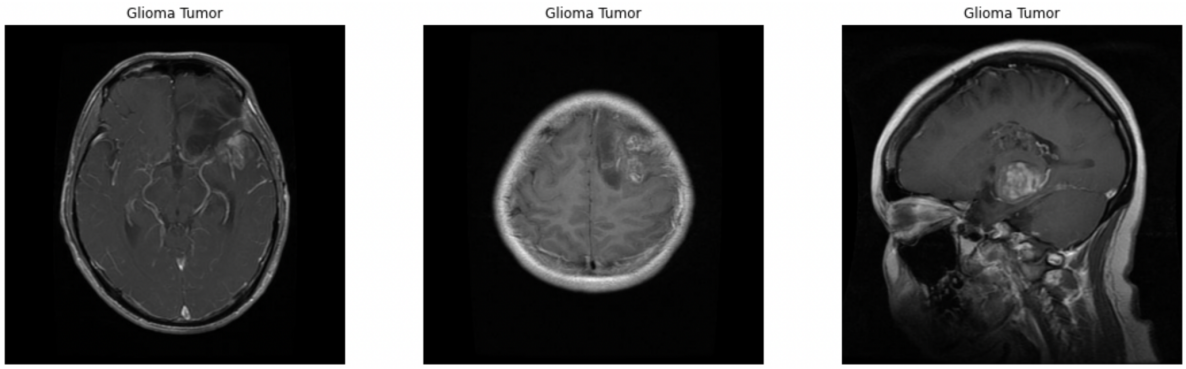



*Figure 1.2 : Brain MRI Images with glioma detected*

Meanwhile, pituitary tumours are tumours that developed from the pituitary gland, a small gland which is located behind the back of the nose and is responsible of producing and releasing hormones that help to carry out vital bodily functions. According to The American Cancer Society (2022), while most pituitary tumours are not cancerous, they can affect the normal functions of the pituitary gland such as making too few or excessive hormones, causing major health problems in the body. As shown below are some of the samples of brain MRI images with pituitary tumours detected.

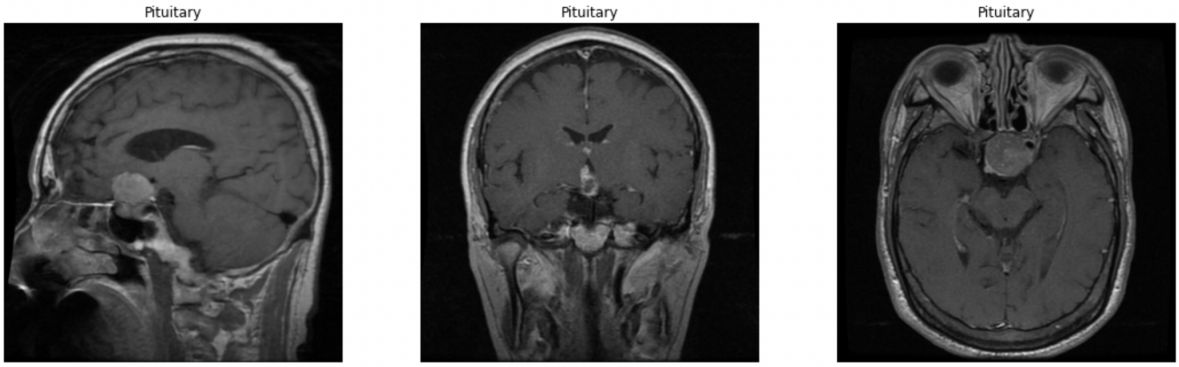



*Figure 1.3: Brain MRI Images with pituitary detected*

## 2.0 Convolutional Neural Network (CNN)

The convolutional neural network can be used in applications such as object recognition, image classification and speech recognition. The CNN is made up of multiple layers of neural network which comprises of mainly three types of layers, namely convolutional layer, pooling layer and fully-connected (FC) layer. As shown in the figure below is the CNN sequence to classify a car.

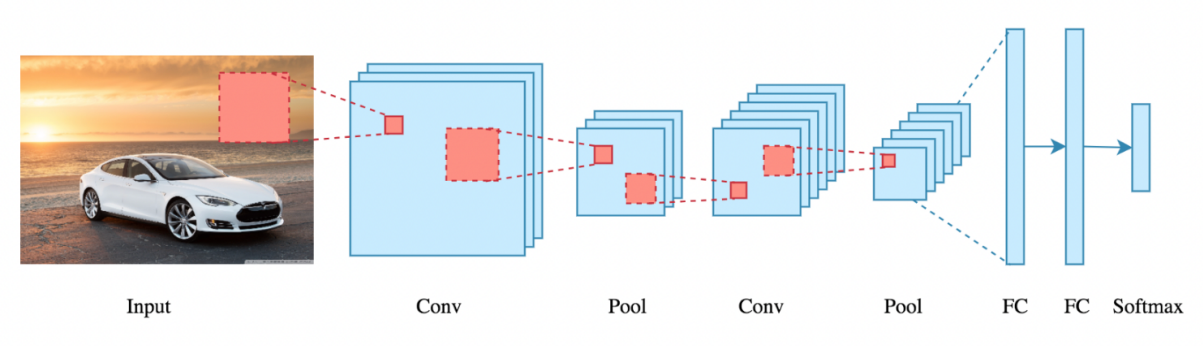



*Figure 2 : Sample of a CNN Architecture to classify an image*

Based on the figure above, the convolutional layer is placed first followed by pooling and then finally fully-connected layers. While the convolutional layers can be followed by another convolutional layers or pooling layers, the fully-connected layers are always the final ones as they are given the task of performing classification (IBM Cloud Education, 2020). The convolutional layers at the front are tasked to recognize simple features e.g. colours and edges which are later used to identify the intended object. Meanwhile, the pooling layer is responsible for dimensionality reduction which helps to decrease the computational power required when processing the data (Saha, 2022). As the image goes deeper through all the CNN layers, more filters are being applied and there will be more features and shapes being recognized until the model finally identifies the intended object.

## 2.1 Transfer Learning

To process and analyse the humongous medical images in the healthcare industry, using deep convolutional neural networks (CNN) models to train the data is time consuming. Hence, pre-trained models which are developed for standard computer vision benchmark datasets are being integrated to speed up the process. This technique is called “Transfer Learning”. Transfer learning is a process where a model developed for a particular task is being re-used in the starting point for another model on another task. Besides speeding up the training process and improving the performance of the model in general, transfer learning is particularly useful when it comes to dealing with insufficient training data because the model is already pre-trained. As shown below is a diagram that depicts the transfer learning approach.

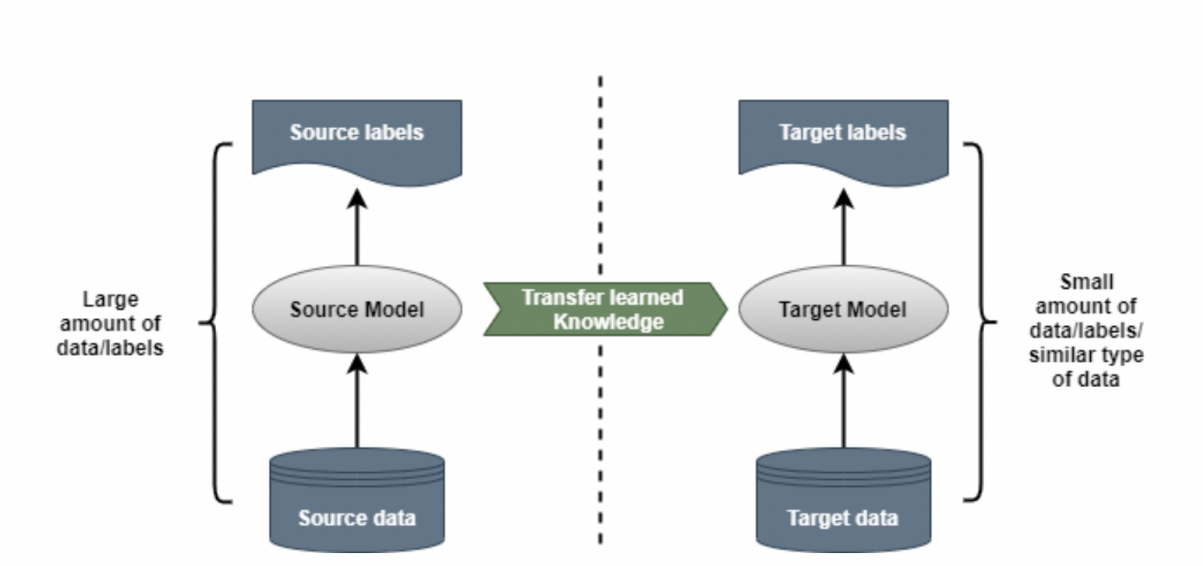



*Figure 2.1 : The transfer learning approach (Mehrotra et al., 2020)*

Based on the diagram, the so-called "knowledge" obtained from a large number of images/data is transferred to a target model. Since the target model is being trained on a small amount of data, the weights, also known as "knowledge" from the pre-trained model can be used to initialize the weights of the target model. This will result in improved efficiency and performance of the target model despite being trained on little data.

There are many pre-trained models which are open-source and openly available to use on the internet. For computer vision problems, some of the popular pre-trained models are VGG-16, Inception V3, Xception and ResNet-50. In this assignment, the pre-trained CNN model architecture, EfficientNet-B0 is used in developing the system to detect and classify brain tumour.

## 2.2 Pre-trained CNN Model

### a) EfficientNet-B0

EfficientNet-B0 is a new baseline network developed by conducting a neural architecture search using the AutoML MNAS framework. The model is the base model from the family of EfficientNet models (B0 to B7). It is trained on more than a million images from the ImageNet database and is able to classify the images into over 1000 categories such as keyboard, mouse, pencil and different types of animals. Besides performing flawlessly on ImageNet, the EfficientNets also achieved satisfactory results on other datasets like the CIFAR-100 and Flowers dataset, with 91.7% and 98.8% accuracy respectively.

## 3. Related Work In Brain Tumour Classification

In recent years, many researchers have ventured into machine learning, specifically deep learning techniques to detect brain tumours and identify them as cancerous or benign. Deep learning architectures are preferred over traditional machine learning classifiers because the former is very robust in terms of feature extraction and less time consuming. In this section, some of the recent literatures related to deep learning-based approaches for identifying and categorizing brain tumours are being discussed.

In Srinivas et al. (2022)'s work, the authors experimented with 3 pretrained CNN models, namely VGG-16, Inception-v3 and ResNet-50 to detect the presence of brain tumour. Their proposed approach is shown in Figure 1 below.

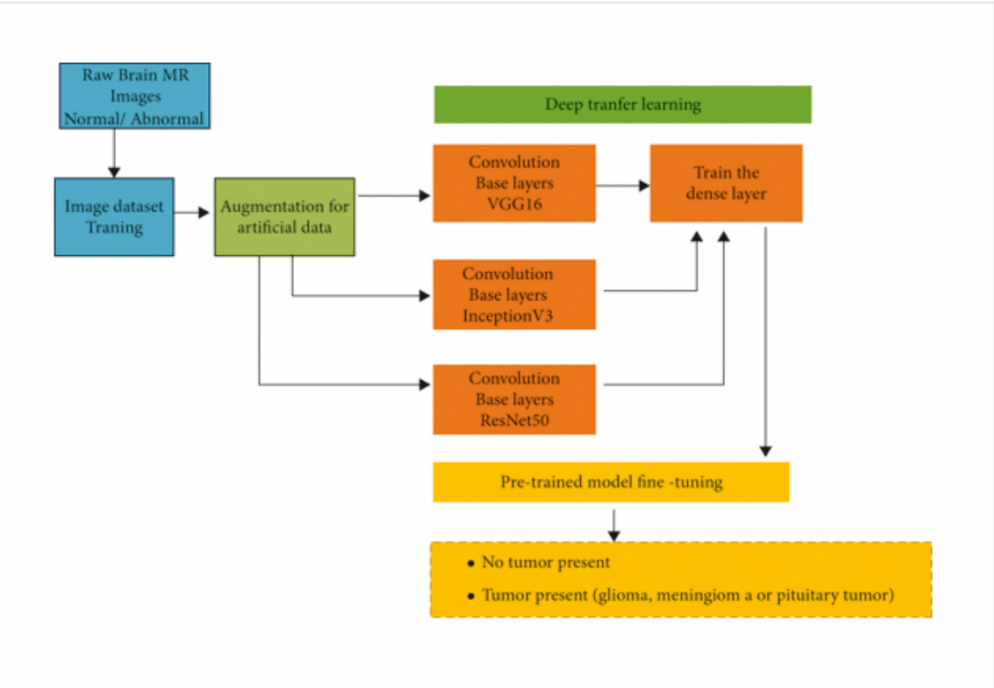



*Figure 3: Proposed system workflow (Srinivas et al., 2022)*

Based on Figure 1, three of the pre-trained models are used as base layers and the training only occurred at the dense layer. They found that all three pre-trained CNN architectures have a training accuracy of more than 90% and validation accuracy of up to 88.26%. Among the pre-trained models, VGG-16 achieved the highest testing accuracy of 96%. During the pre-processing phase, the authors mentioned that it is necessary to resize the brain MRI images into a 224x224x3 dimension before feeding into the pre-trained CNN models because that is the default input size of the model. However, Goel et al., (2022) found no issues when the images were resized to 100x100x3 when feeding into the pre-trained model (ResNet50). The

reason being is because although different pre-trained models come with different input sizes, the input size of the pre-trained models are customizable in Keras (Christos, 2021).

In Sarhan (2020)'s paper, the authors achieved an overall accuracy of 99.3% using their proposed Wavelet-based CNN (WCNN) system. The interesting part about this WCNN system is that unlike most of the brain tumour classification methods which require the segmentation of the input images before feeding into the CNN classifier, their system is able to process the entire brain image with Wavelet decomposition without performing any segmentation to the input images. As shown in Figure 1.2 below is a comparison of the original MRI image with the segmented image.

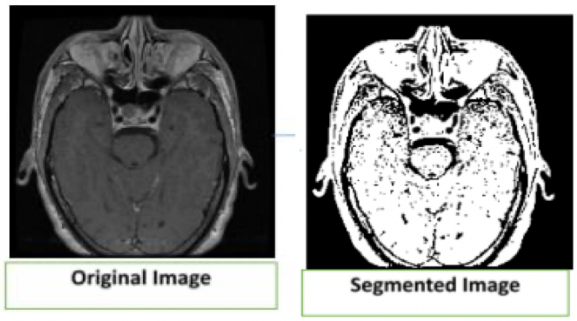



*Figure 3.1: Example of MRI Image before and after segmentation*

The purpose of performing image segmentation is to simplify the processing and analysis of the image. Since the proposed WCNN system processes the images using Wavelet decomposition which can significantly reduce the dimension of the input image, this proposed model has a lower time complexity compared to other systems proposed by other researchers. The Wavelet decomposition highly reduces the dimensions of the input image, which in turn, simplifies the work of the CNN classifier. To improve their model, the author can increase the size of the dataset used to develop the model as this study only utilized 170 images from each class from a total of 3062 of MRI images containing 3 different types of brain tumor, which is considered insufficient to develop a deep learning application.

In Modiya & Vahora (2022)'s research, the authors employed EfficientNet-B7 model with the principal component analysis approach to extract the features of the MRI images. It is found that the combination of features extracted from the EfficientNet model and PCA is able to provide the most significant features for detecting tumours. Their model obtained achieved a training accuracy of 99% and validation accuracy of 80.00 % with learning rate at 0.0001,

batch size at 32 and the maximum number of epochs is 100.

Reza et al. (2022) proposed “DeepTumorNet”, a hybrid deep learning model that can classify different types of brain tumour. Their proposed model used a pre trained model, GoogLeNet, as the base for their CNN architecture. Besides that, the GoogLeNet was modified such that the last 5 layers of the were replaced with 15 new layers which resulted in a total of 154 layers in their proposed hybrid model. As shown in the figure below is the architecture of the proposed “DeepTumorNet” after adding the additional layers.

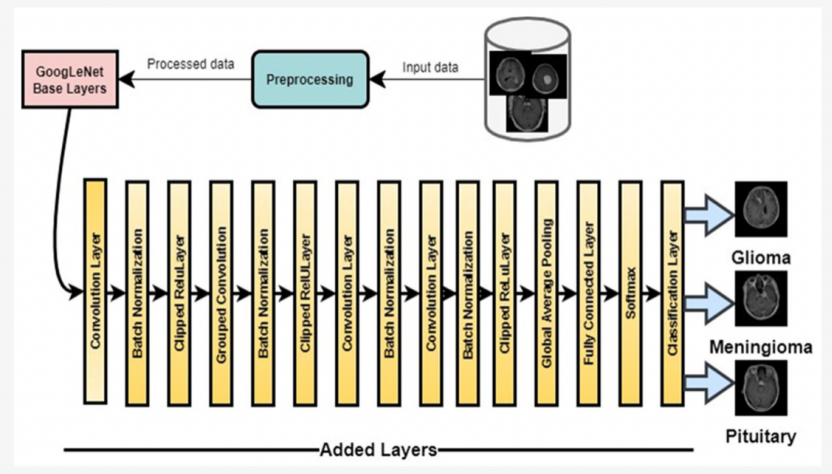



*Figure 3.2: Architecture of the “DeepTumorNet” (Reza et al., 2022)*

Also, a leaky ReLU activation function is used to enhance the expressiveness of the model and to decrease the dying ReLu problem. Their hybrid model achieved a high classification accuracy of 99.67%., which is the highest among other classification methods such as Alex net, Resnet50, darknet53. In their experiment, the authors has concluded that using pre-trained CNN models combined transfer learning techniques will result in the best performance. As shown below is a table summary of the mentioned related works.

| Author and Year | Deep Learning Architecture | Model Performance | Dataset | Recommendations on Future Works |
|---|---|---|---|---|
| Srinivas *et al.* (2022) | Experimented with 3 Pretrained CNN architectures: VGG-16, Inception -v3 and ResNet50 | -All three architectures have a training accuracy of more than 0.90% and validation accuracy up to 0.8826%<br>-VGG-16 gives the best accuracy among the 3 pre-trained CNN architectures | 256 raw MRI images, 158 images are benign tumors while 98 are malignant tumors | Mimic the experiment on other pre-trained CNN models such as VGG-19, MobileNet, and EfficientNet |
| Goel *et al.* (2022) | Resnet50 with learning rate of 0.0001, batch size of 32, 50 epochs and SoftMax activation function | Obtained a training accuracy of 0.9838 and testing accuracy of 0.8794 | 7023 MRI images containing a total number of 4 classes, namely glioma, meningioma, notumor and pituitary | More research should be done on explainable AI to determine understand how the algorithms makes prediction |
| Raza *et al.* (2022) | Proposed 'DeepTumourNet' which is made up of GoogLeNet being used as the base model and the last 5 layers in the GoogLeNet are replaced with 15 new layers<br>-Used leaky ReLU activation function and batch normalisation | Achieved a 99.67% accuracy, 99.6% precision, 100% recall and 99.66% F1-Score | 3064 MRI images containing 3 different types of brain tumor (meningiomas = 708; gliomas = 1426; pituitary = 930) | -To convert other pre-trained CNN models into hybrid models<br>-Check the efficiency of the proposed 'DeepTumourNet' on other medical images e.g. lung cancer, COVID-19 and pneumonia detection |
| Sarhan (2020) | Wavelet-based CNN (WCNN) | The proposed model gives an overall accuracy of 99.3% | 3062 MRI images containing 3 different types of brain tumor (the authors selected only 170 images | - |

| | | | | |
|---|---|---|---|---|
| | | | from each class in his study) | |
| Modiya & Vahora (2022) | -Employed the EfficientNetB7 pre-trained models with Principal Component Analysis (PCA) for feature extraction and then feature reduction<br>-Learning rate is 0.0001, batch size is 32, and maximum number of epochs is 100 | The pre-trained EfficientNet-B7 CNN model with Principal Component Analysis (PCA) achieved 99% training accuracy and 80% validation accuracy | 3000 MRI Images containing 1,500 positive samples and 1,500 negative samples (Data is divided into 80:20 ratio for training and testing) | Use different dimensionality reduction methods |

*Table 1 : Summary of past related works in brain tumour classification*

## Part B: Implementation

## 4. Dataset

The dataset used in this assignment is a combination of a few publicly available brain MRI images dataset. It is obtained from Kaggle and the URL to download the dataset is https://www.kaggle.com/datasets/masoudnickparvar/brain-tumor-mri-dataset. There are two separate files, namely "training" and "testing" provided by the author so that users do not need to perform the train-test split procedure. However, in this assignment, the dataset will be divided manually into training and testing at a ratio of 80:20 because all the images will be combined into a single list when they are imported from google drive. This dataset contains 4 different classes of brain MRI images, namely No Tumor, Glioma, Meningioma and Pituitary. As shown in the table below is the summary if the dataset.

| Dataset | Classes | No. of Images | Total Images |
|---|---|---|---|
| Brain Tumor MRI Dataset | No Tumor | 2,000 | 7,023 |
| | Glioma | 1,621 | |
| | Meningioma | 1,645 | |
| | Pituitary | 1,757 | |

*Table 2: Summary of the Brain Tumor MRI Images*

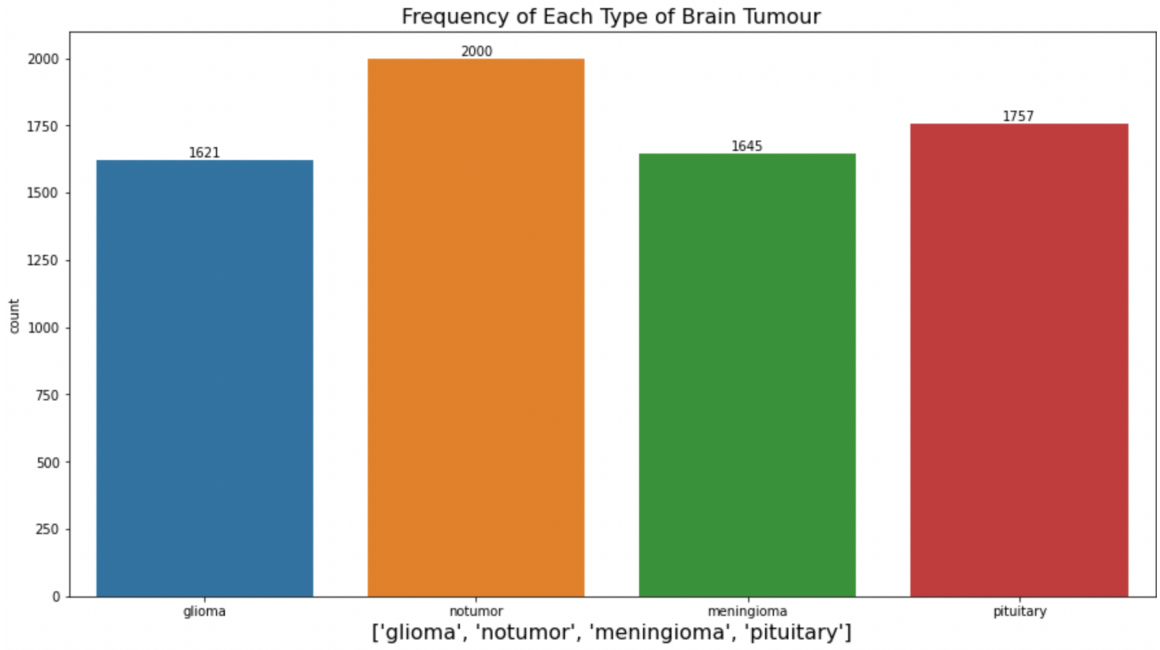
## 5. Data Preparation

```
#We first start off by appending and resize all the images from the directories into a two separate lists,namely x and y.

labels = ['glioma', 'notumor', 'meningioma', 'pituitary']

x = []
y= []
image_size = 224 # all the images will be resized into 150 by 150

for i in labels:
  folderPath = os.path.join('/content/drive/MyDrive/MRI Dataset Version 2 (7023 images)','Training',i)
  for j in tqdm(os.listdir(folderPath)):              #os.listdir will give us all the directories in the 'folderPath'
    img = cv2.imread(os.path.join(folderPath,j))
    img = cv2.resize(img,(image_size, image_size))
    x.append(img)
    y.append(i)

for i in labels:
    folderPath = os.path.join('/content/drive/MyDrive/MRI Dataset Version 2 (7023 images)','Testing',i)
    for j in tqdm(os.listdir(folderPath)):
        img = cv2.imread(os.path.join(folderPath,j))
        img = cv2.resize(img,(image_size,image_size))
        x.append(img)
        y.append(i)

#Then, convert all the resized images and labels into numpy arrays.

x = np.array(x)
y= np.array(y)
```

As shown in the figure above is the codes for data preparation. The dataset downloaded from Kaggle is placed in google drive which can be easily accessible in Google Collab via the “OS” module. During the data preparation phase, to standardize the size of images, all the images in the raw dataset are resized to 224 by 224 as they come in different sizes. The “openCV” python module is used to read and resize the images from the google drive before being stored in both the “x” and “y” list respectively (“x” list contains all the images while “y” list contains all the labels for each respective brain tumor class). Next, both the “x” and “y” list are converted into NumPy arrays. The reason of converting them in to NumPy arrays is to make the training process more efficient as NumPy arrays are more compact and uses less memory.

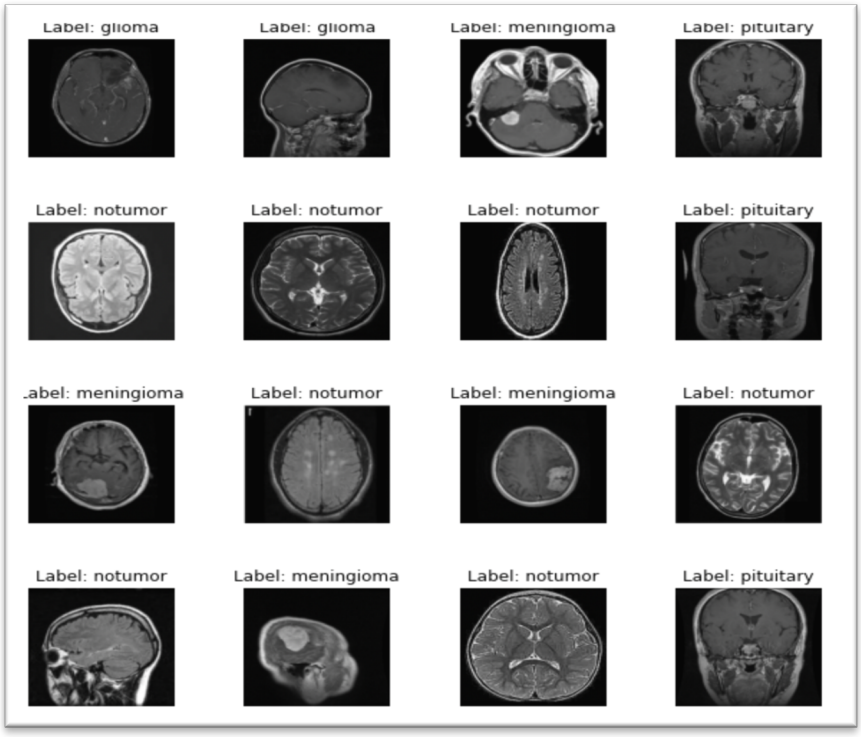
## 6. Exploratory Data Analysis

### 6.1 Bar Chart of Types of Brain Tumour by Count




As shown in the diagram above is the bar chart of frequency of all the labels. There are a total of 7,023 images in the dataset, out of which 1,621 MRI images are labelled as glioma, 2,000 images labelled as “notumour”, 1,645 images labelled as “meningioma” and the remaining 1,757 being labelled as “pituitary”. It is found that the frequency of all four classes is fairly balanced.

## 6.2 Samples of all the Brain MRI Images




The figure above is the type of brain tumor images displayed in a 4x4 grid. The images belonging to individual class is displayed randomly across the grid.

# 7. Data Pre-processing

## 7.1 Data Splitting

```
[23] x_train, x_test, y_train, y_test = train_test_split(x, y, test_size =0.1, random_state=101) # 10
```

Like every other data science related projects particularly in the creation of models based on data, the dataset needs to be divided into two or more subsets to allow for the training of the model and the evaluation/testing of the model. In this assignment, the data is split into 80% training and 20% testing using the “train_test_split” function from Sklearn.

## 7.2 One-Hot Encoding

```
y_train_new = []

for i in y_train:
  y_train_new.append(labels.index(i))

y_train = y_train_new
y_train = tf.keras.utils.to_categorical(y_train)

y_test_new = []

for i in y_test:
  y_test_new.append(labels.index(i))

y_test = y_test_new
y_test = tf.keras.utils.to_categorical(y_test)
```

After splitting the dataset, one hot encoding is conducted on all the labels stored in "y_test" and "y_train". This process will create new binary features for every category of brain tumor in the dataset. The purpose of converting the labels into binary numerical values is to allow the algorithm to understand the data better as it cannot understand the labels in the form of strings.

# 8. Model Implementation

## 8.1 EfficientNet-B0 Implementation: Model 1 (Simple Model)

## a) Constructing the Model 1

```
from keras import layers
from keras.layers import Dense
from keras.optimizers import Adam


no_of_classes = 4                #there are 4 outputs, notumour, glioma, meningioma and pituitary
size = (image_size, image_size)   #image_size was defined earlier as 224

inputs = layers.Input(shape=(image_size, image_size,3)) #"inputs" variable is created so that the EfficientNet model will accept the input as 224

efficient_net1 = EfficientNetB0(
    weights=None,                #weights equals to none because we are not using transfer learning,we are building a custom classifier.
    include_top=True,            #include_top set to True in order to access the last dense layers
    classes=no_of_classes        #number of outputs (4 outputs)
)(inputs)
```

As show in the figure above, a simple model (Model 1) is developed using the EfficientNet-B0 architecture but without the "Imagenet" weights. This means that model is going to be generating all the weights associated with the MRI Images from scratch solely by training the data. Hence, there is no element of transfer learning in this scenario.

## b) Model Architecture

```
model1 = tf.keras.Model(inputs, efficient_net1)

model1.compile(optimizer='adam', loss = 'categorical_crossentropy', metrics=['accuracy'])

print(model1.summary())
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 224, 224, 3)]     0

 efficientnetb0 (Functional)  (None, 4)                4054695

=================================================================
Total params: 4,054,695
Trainable params: 4,012,672
Non-trainable params: 42,023
_________________________________________________________________
None
```

As can be seen from the figure above, the architecture of Model 1 consists of 4,054,695 total parameters whereby 4,012,672 of them are trainable and the remaining are non-trainable parameters. Since this is a simple model, there are a total of 238 layers whereby 237 is the default number of layers of the EfficientNet-B0 and the additional one layer is the input layer which is accepting the images at a size of 224 by 224 by 3.

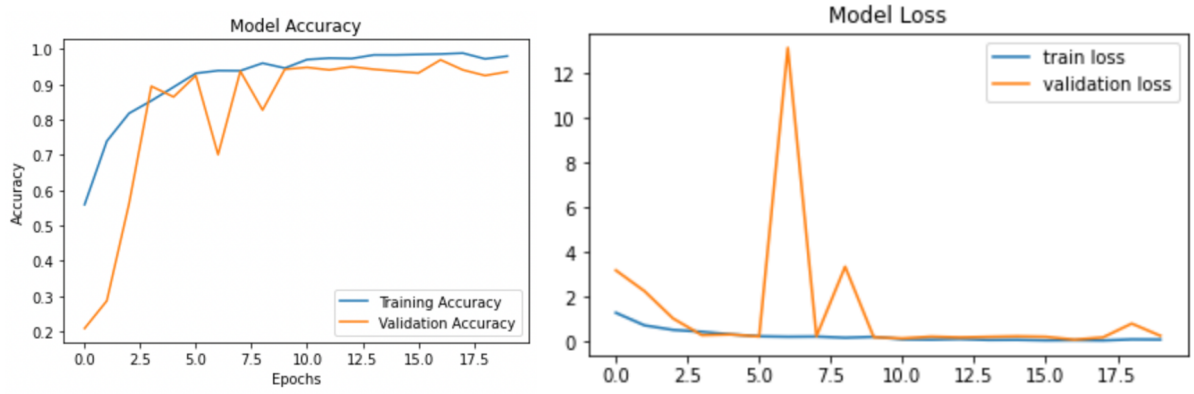
## c) Training Model 1

```
hist1 = model1.fit(x_train, y_train,
                   validation_split = 0.1,
                   epochs=20,
                   verbose = 1,
                   batch_size = 32)
```

During the training of model 1, the model will train the dataset for 20 epochs and 10% of the dataset is set aside for validation purposes. This validation dataset is held back from the training of the model to give an unbiased estimate of the model's performance. Since the batch size is 32, the model will be updated once it has process 32 samples. Also, all the training processes will be stored in the variable called "hist1".

## d) Evaluating the model

```
Epoch 12/20
158/158 [==============================] - 48s 302ms/step - loss: 0.0816 - accuracy: 0.9747 - val_loss: 0.2161 - val_accuracy: 0.9413
Epoch 13/20
158/158 [==============================] - 48s 301ms/step - loss: 0.1073 - accuracy: 0.9735 - val_loss: 0.1815 - val_accuracy: 0.9502
Epoch 14/20
158/158 [==============================] - 47s 300ms/step - loss: 0.0668 - accuracy: 0.9838 - val_loss: 0.2066 - val_accuracy: 0.9431
Epoch 15/20
158/158 [==============================] - 48s 301ms/step - loss: 0.0718 - accuracy: 0.9838 - val_loss: 0.2300 - val_accuracy: 0.9377
Epoch 16/20
158/158 [==============================] - 48s 301ms/step - loss: 0.0478 - accuracy: 0.9852 - val_loss: 0.2085 - val_accuracy: 0.9324
Epoch 17/20
158/158 [==============================] - 47s 300ms/step - loss: 0.0553 - accuracy: 0.9864 - val_loss: 0.0809 - val_accuracy: 0.9698
Epoch 18/20
158/158 [==============================] - 47s 299ms/step - loss: 0.0445 - accuracy: 0.9889 - val_loss: 0.1828 - val_accuracy: 0.9413
Epoch 19/20
158/158 [==============================] - 47s 299ms/step - loss: 0.0925 - accuracy: 0.9725 - val_loss: 0.8002 - val_accuracy: 0.9253
Epoch 20/20
158/158 [==============================] - 48s 302ms/step - loss: 0.0845 - accuracy: 0.9804 - val_loss: 0.2538 - val_accuracy: 0.9359
```




From the figure above, it is found that at the 20th epoch, the training accuracy for Model 1 is 97.45% while the validation accuracy is at 92.53%. For a straightforward/simple model like model 1, the performance of the model is considered satisfactory as the model is a good fit. Due to the fact that the "ImageNet" weights were not included in this model, the model has to generate its own weights from scratch and hence the training accuracy can be seen starting from a relatively low point a (56.15%) in the first epoch.

## e) Testing Accuracy of the Model 1

```
#Using unseen data to test the model
preds = model1.evaluate(x_test, y_test)
print("Loss = " + str(preds[0]))
print("Testing Accuracy of Model 1 is " + str(preds[1]))
```

```
44/44 [==============================] - 3s 60ms/step - loss: 0.1995 - accuracy: 0.9452
Loss = 0.19945448637008667
Testing Accuracy of Model 1 is 0.945195734500885
```

Meanwhile, when feeding model 1 with the test set, the model achieved a testing accuracy of 94.52%.

## 8.2 EfficientNet-B0 Implementation: Model 2 (Tuned Model)

### a) Building Model 2

```
efficient_net2 = EfficientNetB0(weights = 'imagenet', include_top = False, input_shape = (image_size, image_size, 3 ))

#utilize the weights trained on Imagenet on this model
#include_top parameter set to false so that the network doesn't include the top layers/output layer
#this allows us to add our own output layer
```

In model 2, the EfficientNet-B0's pre-trained weights will be transferred to Model 2. When importing the EfficientNet-B0 model, the "include_top" parameter is set to false so that the output layer in the EfficientNet-B0 model will be excluded. This is to allow for the customization of the output layers based on the scenario of the problem. Since the low-level features of the images like edges and blobs are being detected by the pre-trained model (EfficientNet-B0), the top layers will be replaced with additional custom dense layers to recognize higher level features. Meanwhile, the input size of the model will remain the same as the resized MRI images (244 by 244).

### b) Model Architecture

```
# designing the model architecture
model2 = efficient_net2.output
model2 = tf.keras.layers.GlobalAveragePooling2D()(model2)
model2 = tf.keras.layers.Dropout(rate=0.5)(model2)
model2 = tf.keras.layers.Dense(4, activation = 'softmax')(model2)
model2 = tf.keras.models.Model(inputs=efficient_net2.input, outputs = model2)

model2.summary()
```

```
 block7a_project_bn (BatchNorma  (None, 7, 7, 320)   1280        ['block7a_project_conv[0][0]']
 lization)

 top_conv (Conv2D)              (None, 7, 7, 1280)   409600      ['block7a_project_bn[0][0]']

 top_bn (BatchNormalization)    (None, 7, 7, 1280)   5120        ['top_conv[0][0]']

 top_activation (Activation)    (None, 7, 7, 1280)   0           ['top_bn[0][0]']

 global_average_pooling2d (Glob  (None, 1280)        0           ['top_activation[0][0]']
 alAveragePooling2D)

 dropout (Dropout)              (None, 1280)         0           ['global_average_pooling2d[0][0]'
                                                                 ]

 dense (Dense)                  (None, 4)            5124        ['dropout[0][0]']

==================================================================================================
Total params: 4,054,695
Trainable params: 4,012,672
Non-trainable params: 42,023
__________________________________________________________________________________________________
```

```
#show the total number of layers in model 2
print(len(model2.layers))
```

```
241
```

As can be seen from the figure above, the architecture of Model 2 consists of 241 layers whereby 237 layers is the default number of layers in the EfficientNet-B0 and the other 5 additional layers on top of the default layers are our custom layers. The "GlobalAveragePooling2D" is used for applying the pooling operation. This is preferred over the "Flatten" as there is no parameter to optimize in the global average pooling thus overfitting is avoided at this layer. Meanwhile, a dropout layer is added between the pooling layer and the final output later as a regularization technique for the purpose of reducing overfitting.

## c) Model Tuning

```
from keras.optimizers import Adam

# Defining the optimizer, early stopping and learning rate

optimizer = Adam(lr=0.0001)

#early stopping to monitor the validation loss in order to avoid overfitting
early_stop = EarlyStopping(monitor='val_loss', mode = 'min',
                           verbose=1, patience =10, restore_best_weights = True)

#reducing the learning rate on plateau
rlrop = ReduceLROnPlateau(monitor='val_loss', mode = 'min', patience=5 , factor = 0.5, min_lr= 1e-6, verbose =1)

#compiling the model
model2.compile(loss='categorical_crossentropy', optimizer = optimizer, metrics = ['accuracy'])
```

In Model 2, Keras callback functions are used to enhance the model performance. It is a function that is run after each epoch to help with the implementation of early stopping, decaying learning rate. This helps in preventing the model from overtraining the data. Also, the optimizer used is Adam, and the initial learning rate is set to 0.0001.

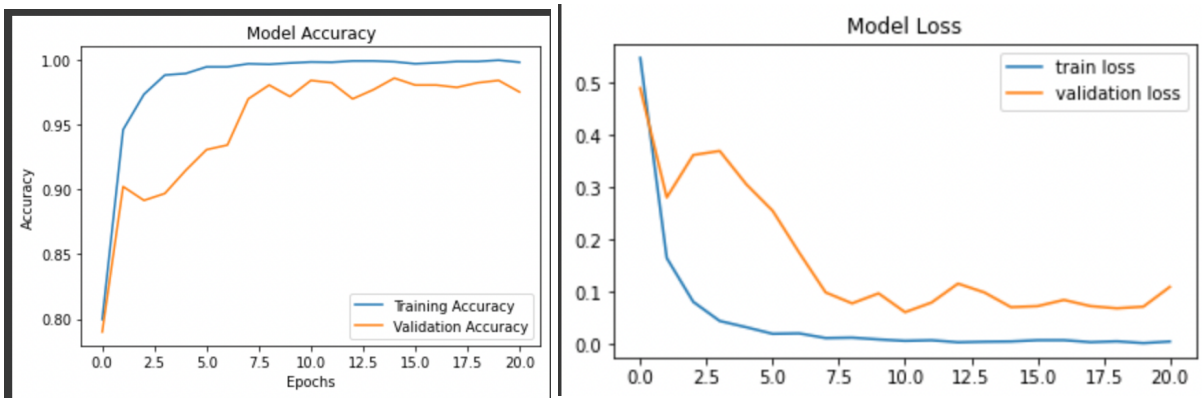
## c) Training the model

```
hist2 = model2.fit(x_train, y_train, validation_split=0.1,
                   callbacks = [early_stop, rlrop] ,      # Pass callback to training
                   epochs = 40 ,
                   verbose = 1,
                   batch_size = 64)
```

During the training of model 2, the model will train the dataset for 40 epochs and 10% of the dataset is set aside for validation purposes. The batch size is increased 64, the model will now be updated once it has process 64 samples. Also, all the training processes will be stored in the variable called "hist2".

## d) Evaluating the model

```
Epoch 14/40
79/79 [==============================] - 45s 568ms/step - loss: 0.0047 - accuracy: 0.9990 - val_loss: 0.0989 - val_accuracy: 0.9769 - lr: 1.0000e-
Epoch 15/40
79/79 [==============================] - 45s 570ms/step - loss: 0.0053 - accuracy: 0.9984 - val_loss: 0.0708 - val_accuracy: 0.9858 - lr: 1.0000e-
Epoch 16/40
79/79 [==============================] - ETA: 0s - loss: 0.0077 - accuracy: 0.9968
Epoch 16: ReduceLROnPlateau reducing learning rate to 4.999999873689376e-05.
79/79 [==============================] - 45s 570ms/step - loss: 0.0077 - accuracy: 0.9968 - val_loss: 0.0728 - val_accuracy: 0.9804 - lr: 1.0000e-
Epoch 17/40
79/79 [==============================] - 45s 569ms/step - loss: 0.0078 - accuracy: 0.9976 - val_loss: 0.0847 - val_accuracy: 0.9804 - lr: 5.0000e-
Epoch 18/40
79/79 [==============================] - 45s 567ms/step - loss: 0.0039 - accuracy: 0.9986 - val_loss: 0.0729 - val_accuracy: 0.9786 - lr: 5.0000e-
Epoch 19/40
79/79 [==============================] - 45s 567ms/step - loss: 0.0055 - accuracy: 0.9986 - val_loss: 0.0686 - val_accuracy: 0.9822 - lr: 5.0000e-
Epoch 20/40
79/79 [==============================] - 45s 574ms/step - loss: 0.0022 - accuracy: 0.9996 - val_loss: 0.0718 - val_accuracy: 0.9840 - lr: 5.0000e-
Epoch 21/40
79/79 [==============================] - ETA: 0s - loss: 0.0052 - accuracy: 0.9980Restoring model weights from the end of the best epoch: 11.

Epoch 21: ReduceLROnPlateau reducing learning rate to 2.499999936844688e-05.
79/79 [==============================] - 45s 571ms/step - loss: 0.0052 - accuracy: 0.9980 - val_loss: 0.1096 - val_accuracy: 0.9751 - lr: 5.0000e-
Epoch 21: early stopping
```




It is found that the model adjusted the learning rate on the 16th epoch and the 21st epoch respectively. The model has stopped training at the 21st epoch as there is no improvement of the model performance on validation dataset is detected beyond the epoch. The final training accuracy achieved for Model 2 is 99.80% while the validation accuracy is at 97.51%.

## e) Testing Model 2's Accuracy

```
#Using unseen data to test the model
preds = model2.evaluate(x_test, y_test)
print("Loss = " + str(preds[0]))
print("Testing Accuracy of Model 2 is " + str(preds[1]))
```

```
44/44 [==============================] - 3s 67ms/step - loss: 0.0196 - accuracy: 0.9943
Loss = 0.01958477310836315
Testing Accuracy of Model 2 is 0.9943060278892517
```

When feeding Model 2 with the test set, the model achieved a testing accuracy of 99.43%.

## 9. Comparison of The Performance of Model 1 and Model 2

### 9.1 Precision, Recall & F1- Score

**Model 1**

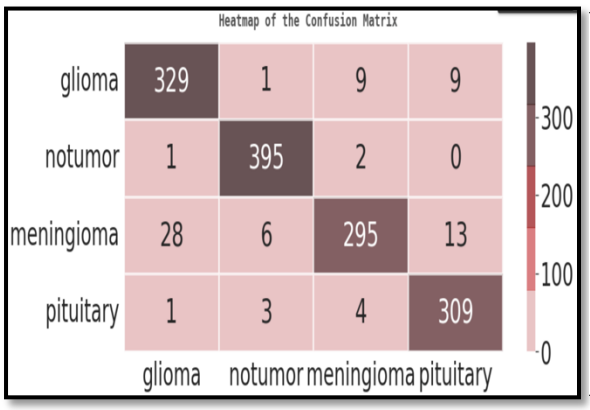
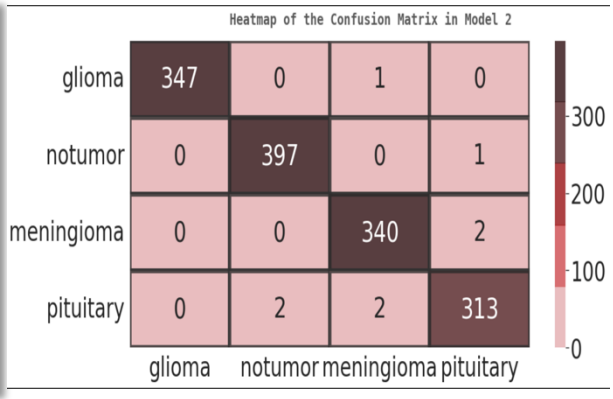
| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.92 | 0.95 | 0.93 | 348 |
| 1 | 0.98 | 0.99 | 0.98 | 398 |
| 2 | 0.95 | 0.86 | 0.90 | 342 |
| 3 | 0.93 | 0.97 | 0.95 | 317 |
| accuracy | | | 0.95 | 1405 |
| macro avg | 0.94 | 0.94 | 0.94 | 1405 |
| weighted avg | 0.95 | 0.95 | 0.94 | 1405 |

**Model 2**

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 1.00 | 1.00 | 1.00 | 348 |
| 1 | 0.99 | 1.00 | 1.00 | 398 |
| 2 | 0.99 | 0.99 | 0.99 | 342 |
| 3 | 0.99 | 0.99 | 0.99 | 317 |
| accuracy | | | 0.99 | 1405 |
| macro avg | 0.99 | 0.99 | 0.99 | 1405 |
| weighted avg | 0.99 | 0.99 | 0.99 | 1405 |

Based on the given metrics, it is found the Model 2 has a better F1-score when detecting and predicting all types of brain tumour. The F1- score combines both the precision and recall to calculate their harmonic mean. Generally, the higher the F1-score, the better the model. When looking at precision, Model 1 seems have quite similar capability in predicting images with no tumour (labelled as 1) with Model 2 because of very close similarity in precision score. Since the use case of these model is to detect if a patient has tumour and the tumour's type, recall should be used to determine the effectiveness of the model because the cost of not detecting the brain tumour or classifying the brain tumour incorrectly (false negative) is high. Hence, Model 2 is a better model because the recall values for all the classes are high (close to 1).

## 9.2 Confusion Matrix

| Model 1 | Model 2 |
|---|---|
|  |  |





A confusion matrix is a two-dimensional table that shows the detailed prediction results of a classification problem. By look at the confusion matrix, the types of errors can be easily identified. Some of the terms used in a confusion matrix are stated as below.

**a) True Positives (TP):** Predicted positive and the actual outcome is positive
**b) False Positives (FP):** Predicted positive and the actual outcome is negative
**c) True Negatives (TN):** Predicted negative and the actual outcome is negative
**d) False Negatives (FN):** Predicted negative and the actual outcome is positive

From the comparison table, it is known that Model 2 has lesser errors compared to the confusion matrix in Model 1. When Model 2 is being tested on unseen data, the model managed to accurately predict 347 images as glioma, 397 as notumor, 340 as meningioma, and 313 images as pituitary. Meanwhile, it has failed to identify the presence of tumour in 2 images. Also, the model has wrongly classified 2 pituitary tumour as meningioma and 2 meningioma tumours as pituitary, 1 glioma image as meningioma and 2 meningioma images as pituitary.

## 10. Summary

| Model | Optimizer | Epochs | Learning Rate | Batch Size | Accuracy | | |
|---|---|---|---|---|---|---|---|
| | | | | | Train Set | Validation Set | Test Set |
| EfficientNet-B0 Model (Without including the pre-trained weights and tuning) | Adam | 20 | 0.001 (Default learning rate for Adam optimizer) | 32 | 98.04% | 93.59% | 94.52% |
| EfficientNet-B0 Model (Utilized Keras callback functions to implement early stopping and decaying learning rate.) | | 40 | Decaying Learning Rate | 64 | 99.8% | 97.51% | 99.43% |

By looking at the table above, Model 2 is better than Model 1 because of the higher accuracy across the training, validation and testing set. Since the application of this model is in the medical field, the model should have a comparatively high accuracy as there is no room for any mistakes. Even with the baseline variant of EfficientNet (B0) and without any tuning performed on it, the model is able to achieve an astonishing testing accuracy of 94.52%. This shows that the EfficientNet CNN architecture can work very well without the hyperparameters being tuned and without the pre-trained weights transferred to it.

Since hyperparameter tuning is also an important aspect of deep learning and can help in increasing accuracy, Model 2 is being developed with the purpose of surpassing Model 1. The different hyperparameters tuned in Model 2 are increasing the number of training epochs, implementing decaying learning rate and increasing the batch size. Although the training process of Model 2 took longer than Model 1, it is still considered to be worth it as there is a slight increase of around 5 % in the testing accuracy. This 5% increment may not seem very

significant on paper, but when comparing it to the number of human lives it can save, this 5% increment suddenly becomes very significant.

In conclusion, experimentation is everything in deep learning as the results may vary with different hyperparameter values and the type of CNN architecture used. The performance of Model 2 (tuned model) is very much likely to improve by using other state of the art versions of EfficientNet e.g. B1 – B7. Also, in the future, it is recommended to train Model 2 with different medical datasets as different datasets may introduce different challenges.

## References


Agrawal, R. K., & Juneja, A. (2019). Deep Learning Models for Medical Image Analysis: Challenges and Future Directions. *Big Data Analytics*, 20–32. https://doi.org/10.1007/978-3-030-37188-3_2

Aldape, K., Brindle, K. M., Chesler, L., Chopra, R., Gajjar, A., Gilbert, M. R., Gottardo, N., Gutmann, D. H., Hargrave, D., Holland, E. C., Jones, D. T. W., Joyce, J. A., Kearns, P., Kieran, M. W., Mellinghoff, I. K., Merchant, M., Pfister, S. M., Pollard, S. M., Ramaswamy, V., . . . Gilbertson, R. J. (2019). Challenges to curing primary brain tumours. *Nature Reviews Clinical Oncology*, *16*(8), 509–520. https://doi.org/10.1038/s41571-019-0177-5

American Association of Neurological Surgeons (AANS). (2022). *Brain Tumors - Classifications, Symptoms, Diagnosis and Treatments*. https://www.aans.org/en/Patients/Neurosurgical-Conditions-and-Treatments/Brain-Tumors

Bhatt, P. S., Jani, C., Shah, D., Ahmed, A., Mariano, M., Singh, H., Graber, J. J., & Salciccioli, J. D. (2022). Trends in incidence and mortality of brain cancer: An observational study of the Global Burden of Disease database from 1990 to 2019. *Journal of Clinical Oncology*, *40*(16_suppl), e18720–e18720. https://doi.org/10.1200/jco.2022.40.16_suppl.e18720

Christos, K. (2021). *Changing input size of pre-trained models in Keras*. Medium. https://ckyrkou.medium.com/changing-input-size-of-pre-trained-models-in-keras-3dfbe3ca3091

*EfficientNet: Improving Accuracy and Efficiency through AutoML and Model Scaling*. (2019, May 29). https://ai.googleblog.com/2019/05/efficientnet-improving-accuracy-and.html

Goel, A. K., Yadav, S., Zubair, & Jain, N. (2022). Artificial Intelligence & Deep Learning: Pet and Spect Imaging and Classification on Brain MRI Images List. *Journal of*

*Physics: Conference Series*, *2325*(1), 012039. https://doi.org/10.1088/1742-6596/2325/1/012039

IBM Cloud Education. (2020). *Convolutional Neural Networks*. https://www.ibm.com/cloud/learn/convolutional-neural-networks

Irmak, E. (2021). Multi-Classification of Brain Tumor MRI Images Using Deep Convolutional Neural Network with Fully Optimized Framework. *Iranian Journal of Science and Technology, Transactions of Electrical Engineering*, *45*(3), 1015–1036. https://doi.org/10.1007/s40998-021-00426-9

Modiya, P., & Vahora, S. (2022). Brain Tumor Detection Using Transfer Learning with Dimensionality Reduction Method. International Journal of Intelligent Systems and Applications in Engineering, 10(2), 201–206. Retrieved from https://ijisae.org/index.php/IJISAE/article/view/1310

Mehrotra, R., Ansari, M., Agrawal, R., & Anand, R. (2020). A Transfer Learning approach for AI-based classification of brain tumors. *Machine Learning With Applications*, *2*, 100003. https://doi.org/10.1016/j.mlwa.2020.100003

National Health Service. (2021). *Brain tumours*. nhs.uk. https://www.nhs.uk/conditions/brain-tumours/

Park, S. H., Won, J., Kim, S. I., Lee, Y., Park, C. K., Kim, S. K., & Choi, S. H. (2017). Molecular Testing of Brain Tumor. *Journal of Pathology and Translational Medicine*, *51*(3), 205–223. https://doi.org/10.4132/jptm.2017.03.08

Raza, A., Ayub, H., Khan, J. A., Ahmad, I., S. Salama, A., Daradkeh, Y. I., Javeed, D., Ur Rehman, A., & Hamam, H. (2022). A Hybrid Deep Learning-Based Approach for Brain Tumor Classification. *Electronics*, *11*(7), 1146. https://doi.org/10.3390/electronics11071146

Saha, S. (2022). *A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way*. Medium. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Sarhan, A. M. (2020). Brain Tumor Classification in Magnetic Resonance Images Using Deep Learning and Wavelet Transform. *Journal of Biomedical Science and Engineering*, *13*(06), 102–112. https://doi.org/10.4236/jbise.2020.136010

Srinivas, C., K. S., N. P., Zakariah, M., Alothaibi, Y. A., Shaukat, K., Partibane, B., & Awal, H. (2022). Deep Transfer Learning Approaches in Performance Analysis of Brain Tumor Classification Using MRI Images. *Journal of Healthcare Engineering*, *2022*, 1–17. https://doi.org/10.1155/2022/3264367

The American Cancer Society. (2022). *What is a Pituitary Tumor? | American Cancer Society*. https://www.cancer.org/cancer/pituitary-tumors/about/what-is-pituitary-tumor.html

Tsuneki, M. (2022). Deep learning models in medical image analysis. *Journal of Oral Biosciences*, *64*(3), 312–320. https://doi.org/10.1016/j.job.2022.03.003